1994

A competitive learning approach for solving the capacitated location-allocation problem

Jennifer S. Chuss
Lehigh University

AUTHOR:

Chuss, Jennifer S.

TITLE:

A Competitive Learning Approach for Solving the Capacitated Location-Allocation Problem

DATE: October 9, 1994

A Competitive Learning Approach for Solving the Capacitated Location-Allocation Problem

by

Jennifer S. Chuss

A Thesis

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Industrial Engineering

Lehigh University

October 9, 1994

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

9/22/94
Date

Thesis Advisor

Chairperson of Department

Table of Contents

List of Tables

List of Tables Cont'd

List of Figures

Abstract

This paper presents findings from using a neural network approach to solve the Euclidean facility location problem. CLOC (Competitive LOCation) is an adaptive neural method, with a slightly modified weight update, developed by Burke to solve the Euclidean problem. CLOC performs favorably when compared with an existing heuristic method developed by Koskosidis and Powell for solving the Capacitated Clustering Problem (CCP). It is an extremely fast algorithm that seems to be computationally superior to Koskosidis and Powell's heuristic. CLOC appears to be a candidate solution approach for these capacitated warehouse location-allocation problems, especially real-world problems that are not random in nature. CLOC has a tendency to not use all available facilities, which is a desirable feature when considering there are costs associated with using or building a facility.

Introduction

A company has various customers located across the country who demand products. Where should the company build or lease warehouses? Which warehouse should service each customer so that costs are minimized? To complicate the problem further, warehouses have limited capacity. Total customer demand at a warehouse must not exceed its capacity. This problem is called the capacitated warehouse location-allocation problem. Such problems are both integer and nonlinear and thus are very difficult to solve.

At Air Products and Chemicals, Inc., the warehouse locations have already been established by a less formal method (i.e. without a mathematical model). These locations are frequently reviewed for appropriateness. The problem then becomes one of assigning customers to warehouse locations. This type of problem is typically solved using a linear program. Assumptions are made about the costs so that they can be modeled linearly and the integrality constraints are relaxed. However, as these problems become larger and larger, it becomes less practical to solve them using a linear programming approach. The need for determining warehouse locations at Air Products is not encountered often, but it is explored in this paper because it is a problem of general interest.

The purpose of this work is to determine the feasibility of using CLOC (Competitive LOCation), an adaptive neural method with a slightly modified weight update developed by Burke (Burke, 1994), to solve the capacitated location-allocation problem. In order to decide if using CLOC is feasible, the neural network algorithm will be compared with a heuristic approach developed by Koskosidis and Powell (Koskosidis and Powell, 1991). In testing a variety of problems, they have found very good results. The two

approaches will be compared on 1) a simple measure - time to solve and 2) the quality of the results - total distance, tightness of the clusters and the number of clusters formed.

2.0 Koskosidis and Powell's Heuristic Discussion

Koskosidis and Powell have developed a heuristic for solving the Capacitated Clustering Problem (CCP) (Koskosidis and Powell, 1991). It will be referred to as "KP" throughout this paper. As the authors discuss in their paper, the idea behind the CCP is to partition a set of N customers into M mutually exclusive groups. As part of the process, clustering points or "seeds" for these groups are determined. Also, the size of the group is restricted, where size usually refers to a capacity.

One objective of the CCP is to form groups to minimize distance within groups and maximize distance between groups while obeying capacity restrictions. Another objective is just to minimize distance within groups while also meeting capacity restrictions. The latter objective is the one of interest in this research.

KP was developed with the idea of consolidating customer orders into full-truck shipments. However, as Koskosidis and Powell discuss, many applications require grouping customers while meeting size restrictions of the group and trying to minimize total distance within all groups. Their heuristic is applicable for these problems as well. The capacitated warehouse location-allocation problem is such a problem. KP will be used to form clusters of customers to assign to facilities such that capacity of the facilities is not violated, while minimizing total distance within all clusters.

There are 3 main steps in Koskosidis and Powell's approach. They can be summarized as follows: Start with an initial set of seeds. The first step is to form clusters by assigning customers to the "closest feasible seed". The "closest" seed is the seed with the smallest cost to the customer. Koskosidis and Powell pre-define this cost. In this paper, the closest seed is the seed with the minimum Euclidean distance to the customer. A "feasible" seed is one with sufficient capacity to handle the customer's demand.

Next, improve the seed location within each cluster. In other words, choose a customer as the seed to minimize the distance within the cluster. (Note that seeds are actual customer locations in this heuristic.) Finally, perform local exchanges. By swapping customers between clusters, can the total distance be improved?

In their paper, Koskosidis and Powell discuss various methods for initializing the seeds. Instead of simply using randomly-generated seeds, they suggest approaches that are likened to set covering algorithms. Instead of following one of these approaches, a competitive learning neural network is used to initialize the seeds in this research. Potvin describes such a seed initialization approach in his paper (Potvin, 1994) as a precursor to a vehicle routing and scheduling algorithm.

In later discussions of CLOC, it will become apparent that this is a reasonable approach. The competitive learning seed initialization neural network differs from CLOC in 2 ways: 1) CLOC has a slightly different weight update to handle the Euclidean problem and 2) the competitive learning network is uncapacitated, whereas in CLOC, capacities are an issue. Though the Euclidean problem and the clustering problem <u>are</u> different, this

seed initialization should produce a reasonable starting point since it will produce the centroids of clusters of customers.

3.0 KP Implementation

The next sections describe the implementation of Koskosidis and Powell's heuristic along with details of the competitive learning seed initialization procedure used. Test problems are then described and finally, results of the analyses are discussed.

For the seed initialization phase, a competitive learning program (CLVRP) written by Burke, coded in Microsoft FORTRAN was used. A diagram of the network appears in Figure 1. This structure, with the inclusion of an extra node at the input layer, is necessary for the neural network code to be able to find the output node having the minimum Euclidean distance to the input pattern, as opposed to finding an output node that is most similar to the input pattern. Otherwise, all inputs and weight vectors would have to be normalized to constant length, which causes the loss of all geographical features of the data. The steps in the competitive learning seed initialization procedure are as follows:

1) Initialize the weight vector for all output nodes where the number of output nodes is fixed *a priori*. Values at which to initialize the weight vectors will be discussed later.
2) Initialize other parameters, such as learning rate, decay in learning rate at each cycle and the number of cycles. Values at which to intelligently initialize these parameters will also be discussed later.

3) Read an input pattern (X_1, X_2, -1) where X_1 is customer latitude, X_2 is customer longitude and -1 is what enables the transmission of distance.

4) Winning output node J is node with greatest net input where net input is defined as:

$$net_J = \Sigma_i X_i W_{iJ} - \Sigma_i (W_{iJ}^2) / 2$$

5) Adjust J's weights by:

$$W_{iJ}^{new} = W_{iJ}^{old} + \eta(X_{iJ} - W_{iJ}^{old})$$

where η is a learning rate, typically initialized close to 1 and gradually decreased over successive iterations.

Continue reading patterns and adjusting weights until all patterns are read through. Then, update η as η * decay term. Continue as above, reducing the learning rate (η) after each cycle until the program has gone through the requested number of cycles.

Koskosidis and Powell's algorithm was coded using Microsoft FORTRAN as well. For the definition of cost between two locations, Euclidean distance is used whereas KP assumes a pre-defined cost matrix. The major steps in their algorithm appear below:

(Step 0) Start with an initial set of seeds

(Step 1.1) For each customer i, determine C_{ij}, the Euclidean distance between it and all seeds j. Sort seeds in increasing order of C_{ij}. Calculate regret value for each customer where regret(i) = $C_{ij}^2 - C_{ij}^1$. j^1 is the closest seed and j^2 is the next closest seed.

(Step 1.2) Sort all regret(i) in decreasing order. Assign each customer, in decreasing order of regret, to its closest available seed, if the capacity is available. If capacity isn't available, assign to next closest seed and so on. If no seed has enough capacity, go to step 1.3. If all customers assigned, go to Step 2.1.

(Step 1.3) If, in Step 1.2, there wasn't sufficient capacity for customer i, turn customer i into a seed. Return to Step 1.1 with one seed more than previous time through.

(Step 2.1) For a particular cluster, calculate the total distance from the candidate seed to each customer in the cluster. Do this for all customers in turn acting as the candidate seed. Find the minimum such distance within the cluster. The associated candidate seed becomes the new seed. Repeat this step for all clusters.

(Step 2.2) If old cluster distance - new cluster distance > ε, go to Step 1. Otherwise, go to Step 3.

(Step 3.1) Pick any pair of clusters, k_1 and k_2 with seeds j_1 and j_2 respectively. If all pairs considered, go to Step 3.4.

(Step 3.2) Pick any pair of customers, one from k_1 and one from k_2. If all pairs considered, go to Step 3.1.

(Step 3.3) Calculate ΔC as the "cost" of exchanging the 2 customers. If $\Delta C \geq 0$ (meaning no improvement in cluster distance), go to Step 3.2. Otherwise, check the capacity feasibility constraints. If no location's capacity violated, then swap the 2 customers between their respective clusters.

(Step 3.4) If any local exchanges have been made, go to Step 2. Otherwise Stop.

A copy of the code for the seed initialization procedure and KP can be found with Dr. Burke in the Industrial Engineering Department.

4.0 KP Test Problems

The test problems used for testing both KP and CLOC are of two types: 1) randomly-generated test problems and 2) actual customer data from Air Products. To test both algorithms' ability to handle different sized problems, 5 test cases each of 50, 75 and 100 customers were generated. Because of the random nature of the problems, customer latitude and longitude are evenly distributed along the specified ranges. Demand is evenly distributed as well -- between 1 and 50 pounds. (See Tables 1, 2 and 3.)

Table 1. Test Data - 50 customers

Case #	1A	1B	1C	1D	1E
Lat Range	5 - 63	2 - 70	8 - 69	3 - 68	3 - 69
Lat Median	36.5	31.5	37.0	43.0	38.0
Lat Mean	35.0	35.4	37.0	36.2	35.9
Long Range	6 - 69	5 - 80	4 - 78	5 - 80	11 - 78
Long Median	38.5	41.0	42.0	49.5	46.0
Long Mean	39.0	39.0	42.7	45.9	44.7
Demand Range	3 - 41	4 - 47	1 - 49	2 - 50	5 - 50
Demand Median	15.0	26.0	22.5	30.0	31.0
Demand Total	777	1169	1200	1354	1432
Demand Mean	15.5	23.4	24.0	27.1	28.6

Table 2. Test Data - 75 customers

Case #	2A	2B	2C	2D	2E
Lat Range	6 - 70	4 - 68	3 - 69	3 - 70	3 - 70
Lat Median	40	30	38	44	31
Lat Mean	39.3	32.3	37.3	40.3	32.6
Long Range	4 - 76	3 - 80	3 - 80	3 - 80	4 - 80
Long Median	36	45	43	43	45
Long Mean	36.7	43.8	41.3	38.1	43.1
Demand Range	1 - 37	2 - 50	1 - 50	2 - 50	1 - 49
Demand Median	18	30	25	27	20
Demand Total	1364	2165	1870	2043	1738
Demand Mean	18.2	28.9	24.9	27.2	23.2

Table 3. Test Data - 100 customers

Case #	3A	3B	3C	3D	3E
Lat Range	2 - 67	3 - 70	3 - 70	2 - 70	3 - 70
Lat Median	31.0	30.5	31.0	38.5	36.0
Lat Mean	33.7	32.9	32.2	38.4	35.5
Long Range	3 - 77	4 - 80	3 - 80	3 - 80	3 - 80
Long Median	34.5	46.0	43.0	46.0	38.5
Long Mean	35.9	43.5	43.3	43.7	39.9
Demand Range	1 - 41	1 - 50	1 - 50	1 - 50	1 - 50
Demand Median	13.0	28.5	26.0	26.5	22.0
Demand Total	1458	2727	2625	2596	2317
Demand Mean	14.6	27.3	26.3	26.0	23.2

The Air Products test cases are larger, with 300 customers. Because this is actual data, customers are more densely located in some areas as opposed to others. The demand is not uniformly distributed within a given range either. Demand values for a customer can range from 100 pounds to over a million pounds. These 5 test cases will help determine both algorithms' ability to handle larger problems that are not random in nature. (See Table 4.)

Table 4. Test Data - 300 customers

Case #	APCIa	APCIb	APCIc	APCId	APCIe
Lat Range	26 - 42	26 - 46	26 - 42	26 - 37	26 - 40
Lat Median	35.7	39.2	37.0	34.0	35.16
Lat Mean	35.8	37.6	36.8	33.5	35.30
Long Range	73-122	70-123	71-122	76-122	75-122
Long Median	85.0	82.6	86.1	86.8	86.1
Long Mean	91.1	86.1	90.5	92.3	92.1
Demand Range	0 - 332	0 - 1144	0 - 24	0 - 252	0 - 1053
Demand Median	1.0	2.0	0.0	1.0	2.0
Demand Total	2636	7841	557	3219	7623
Demand Mean	9.0	26.1	1.9	10.7	25.4

In total, there are 15 randomly-generated test cases and 5 test cases with actual data. All sets of data are used throughout the analyses. The first goal is to determine what "parameters" in the seed initialization procedure of KP and which parameters in CLOC produce the best final results. Several parameters are available for tuning in the seed initialization procedure chosen for KP:

1) learning rate and learning rate decay
2) number of output nodes and the capacity on those nodes
3) order of presentation of input patterns
4) initial weight vectors

These parameters do not directly affect KP since they are not part of KP. However, because of their effect on the centroids of the clusters formed by CLVRP, these parameters will all have some impact on the final warehouse locations and customer allocations formed by KP.

4.1 Learning Rate and Learning Rate Decay

In her research, Burke has found that an initial learning rate close to 1 and a geometric decrease in learning rate close to 1 after each cycle provides good results. Does the same thing apply to these problems or does a linear decrease work as well or better? The following scenarios were examined:

1) initial learning rate of 1.0, geometric decrease after each cycle of 0.99, 100 cycles
2) initial learning rate of 1.0, linear decrease after each cycle of 0.01, 100 cycles
3) initial learning rate of 1.0, linear decrease after each cycle of 0.01, 50 cycles

For this analysis, the number of cycles also comes into play. For a geometric decrease in learning rate, Burke (Burke, 1994) has estimated the number of cycles to be: $M(\log \varepsilon) / N(\log dec)$ where M is the number of facilities, N is the number of customers, ε is a small constant and *dec* is the decay in the learning rate after each cycle. Using this estimate, 100 cycles is sufficient for the different-sized test cases.

For a linear decrease in learning rate, it is apparent that an initial learning rate of 1 with a decay term of 0.1 will not show any adjustment in the weight vector after 10 cycles since the learning rate has been reduced to 0. The number of cycles is highly-dependent on the initial learning rate, the decay in learning rate and whether it's a geometric or linear decay. Thus, the number of cycles used is chosen accordingly.

4.2 Number of Output Nodes and Capacities on those Nodes

Another consideration is the number of output nodes and the capacity on those nodes. The capacities for the test problems have been fixed and the

number of output nodes (or warehouses) is allowed to fluctuate. For the randomly-generated problems, a capacity of 250 pounds is used. For the Air Products problems, the capacity is 1.2 million pounds. These capacities were chosen so that the required number of output nodes is manageable. Two scenarios were used in examining the effect of the number of output nodes on the solution:

1) the minimum number of output nodes required
2) more-than-enough output nodes

In this analysis, scenario 2) above will always produce better results than scenario 1); allowing more output nodes will cause tighter clusters to be formed. These two scenarios will not be compared to each other. Rather, the effect the total demand to total capacity ratio has on the number of output nodes used and any differences between KP's and CLOC's handling of the two scenarios will be examined.

4.3 Order of Input Pattern Presentation

The effect of presenting the customer records in different orders -- random order, increasing order of demand and decreasing order of demand -- will be analyzed. What effect does ordering the demand differently have on 1) the final solution and 2) the number of output nodes formed, especially in the tightly-constrained scenario?

4.4 Initial Weight Vectors

Also of importance are the initial weights that are assigned to each of the output nodes. These weight vectors need to be chosen carefully. As Burke discusses in her paper (Burke, 1994), the initial weight vector indicates the relative cost of opening a new facility. If a customer is closer to

an initial weight vector than to any other facilities, a new facility is opened. For these analyses, the following initial weight vectors were used:

1) All output nodes initialized to the same weight vector -- the median latitude and longitude of all customer locations
2) Output nodes divided into 4 sets where the sets are: high/high, high/low, low/high, low/low. High/high indicates the latitude and longitude are both relatively high, high/low means the latitude is relatively high, but the longitude is relatively low, etc.
3) For the Air Products data only, initial weight vectors in the area where customers are more densely located were tested.

5.0 KP Computational Results

In the following sections, details from testing scenarios for the various parameters described above are discussed. To determine the relevance of the parameter at-hand, other parameters are held fixed. Therefore, the parameters fixed and the values at which they are fixed are stated. Results are also discussed, in terms of what value for each parameter worked best, as well as general observations made while testing that particular parameter. The same type of results will be provided later for CLOC as well as a comparison of the final results for KP and CLOC.

5.1 Learning Rate and Learning Rate Decay

In determining the "best" learning rate and learning rate decay to use for the seed initialization procedure (CLVRP) of KP, the other parameters were fixed at the following values:

- number of output nodes for 50-customer cases: 5
- number of output nodes for 75-customer cases: 8
- number of output nodes for 100 customer cases: 10

- number of output nodes for 300 customer cases: 10
- capacity on each output node: equal to total demand
- initial weights on output nodes: equal to median of customer latitudes and longitudes

For the learning rate analysis, the capacitated problem has not yet been considered. Although capacities are included in the test cases, the number of output nodes and the capacities on those output nodes are set in such a way that the problem is essentially uncapacitated. The number of output nodes varies by case type where case type is defined by the number of customers; 50, 75, 100 and 300. The initial weights for each test case are all set to the same value -- the median of the customer latitudes and longitudes in that test case.

As a starting point, KP results are compared with a k-means clustering algorithm as a benchmark. The FASTCLUS procedure in SAS, which minimizes the sum of squared Euclidean distances from cluster means to observations in the cluster, was used. The number of clusters desired is specified. Since FASTCLUS does not consider facility capacities and customer demands, this comparison can only be made for the uncapacitated problem. The average total Euclidean distance for each problem type (50 customers, 75 customers, etc.) and the average of the maximum cluster distance are displayed in Table 5 for k-means and the three learning rate scenarios.

Table 5. Comparison of k-means clustering and KP with various values of learning rate and learning rate decay for the seed initialization procedure*.

Number of Custs	k-means clustering	1.0 initial η *0.99 decay 100 cycles	1.0 initial η -0.01 decay 100 cycles	1.0 initial η -0.01 decay 50 cycles
50	564 / 186	519 / 178	528 / 165	531 / 162
75	691 / 149	659 / 145	669 / 148	672 / 147
100	849 / 155	778 / 141	765 / 126	800 / 138
300	679 / 216	613 / 185	586 / 176	590 / 156

* Results represent average of total Euclidean distance / average of maximum cluster distance for each problem type.

Although all summarized results for KP are better than k-means (both total Euclidean distance and maximum cluster distance), they are not significantly different, except for the 100-customer problems. Also, there isn't any statistical difference between KP using the geometric decrease in learning rate versus the two scenarios using a linear decrease.

However, the individual case results showed that 100% of the 20 cases using KP with a geometric decrease in learning rate had better results than k-means, 90% of the cases using a linear decrease with 100 cycles had better results than k-means and 95% of the cases using a linear decrease with 50 cycles had better results than k-means. When comparing the three scenarios for KP with each other on an individual case basis, results from KP with a geometric decrease in learning rate are best in 65% of the cases. Though the results are not significantly different, the majority of cases are better with a geometric decrease in learning rate. Therefore, a geometric decrease in learning rate will be used for the seed initialization procedure of KP in future analyses.

Also, an observation that was made is that sometimes the exact same results are produced by the different learning rate scenarios. Using different

learning rate decays will cause the seed initialization procedure to produce different centroids of clusters, but they are not so dissimilar that KP will generate different facilities and associated customer assignments. KP uses the initial seeds as a starting point. In Step 2.1 it replaces the current seed with a customer that, when acting as a seed, produces the minimum distance in the cluster. Two seeds from the two scenarios may be different. But customer x may turn out to always be the best seed for that cluster.

5.2 Number of Output Nodes and Capacities on those Nodes

In analyzing the number of output nodes to use in the seed initialization procedure of KP, the other parameters were fixed at the following values:

- capacity for random cases (50, 75 and 100 customers): 250 lbs
- capacity for APCI cases (300 customers): 1,200,000 lbs
- initial weights on output nodes: equal to median of customer latitudes and longitudes
- initial learning rate: 1.0
- learning rate decay: geometric (0.99)
- number of cycles: 100

The two scenarios of interest are a tightly-constrained case and a less-tightly-constrained case. The total demand to total capacity ratio is what determines how tightly the problem is constrained. In Table 6, the minimum number of output nodes required to meet all customers' demand has been determined. This number varies by case. The total demand to total capacity ratios will be high for this tightly-constrained scenario.

Table 6. Demand Analysis

Case #	Demand Range	Min. Cap. / Node	Cap.	Total Demand	# Output Nodes Req'd
1A	3 - 41	41	250	777	3+
1B	4 - 47	47	250	1169	4+
1C	1 - 49	49	250	1200	4+
1D	2 - 50	50	250	1354	5+
1E	5 - 50	50	250	1432	5+
2A	1 - 37	37	250	1364	5+
2B	2 - 50	50	250	2165	8+
2C	1 - 50	50	250	1870	7+
2D	2 - 50	50	250	2043	8+
2E	1 - 49	49	250	1738	6+
3A	1 - 41	41	250	1458	5+
3B	1 - 50	50	250	2727	10+
3C	1 - 50	50	250	2625	10+
3D	1 - 50	50	250	2596	10+
3E	1 - 50	50	250	2317	9+
APCIa	0 - 332	332	1200	2636	2+
APCIb	0 - 1144	1144	1200	7841	6+
APCIc	0 - 24	24	1200	557	0+
APCId	0 - 252	252	1200	3219	2+
APCIe	0 - 153	153	1200	7623	6+

For the second scenario, 13 output nodes (or facilities) are allowed. However, KP may determine that additional facilities are required. If so, it will create them (Step 1.3 in the algorithm). For this scenario, the total demand to total capacity ratio will be lower. The average total Euclidean distance for each problem type and the average of the maximum cluster distance are displayed in Table 7 for the two scenarios, along with the range for the total demand to total capacity ratio.

Table 7. Comparison of tightly-constrained case (Scenario 1) and loosely-constrained case (Scenario 2) based on total demand to total capacity ratio*.

# of Custs	Scenario 1	Ratio for Clusters Spec.	Ratio for Clusters Formed	Scenario 2	Ratio for Clusters Formed
50	560 / 154	0.78 - 0.96	0.78 - 0.95	268 / 51	0.24 - 0.44
75	700 / 120	0.91 - 0.99	0.83 - 0.91	477 / 67	0.42 - 0.67
100	905 / 131	0.93 - 0.99	0.83 - 0.95	688 / 96	0.45 - 0.84
300	1563 / 1057	0.46 - 0.93	0.46 - 0.89	561 / 157	0.04 - 0.50

* Results represent average of total Euclidean distance / average of maximum cluster distance for each problem type.

Obviously, the results for the loosely-constrained case will be better than the tightly-constrained case since there are more facilities over which to spread the customers' demand, so tighter clusters will be formed. For the cases where the minimum number of output nodes is specified, there was not always enough capacity to satisfy the demand, so more facilities were added. There may have been sufficient capacity overall, but there was not enough capacity in each facility to provide for single-sourcing of customer demand.

The range of total demand to total capacity ratios for the number of facilities specified, and the ratio for the number of facilities actually formed is indicated in the table. For the randomly-generated cases (50, 75 and 100 customers), new facilities are added when the ratio is roughly 94% and above. For the APCI cases, new facilities are added when the ratio is roughly 91% and above -- a bit lower than the random cases.

The APCI cases are different because they are not random. For instance, there is great variability in the demand values among the APCI cases. In one case, there are roughly 6% of the customers with over 60% of the demand. In addition, one APCI case has total demand of 557,000 pounds, while another case has total demand over 7.8 million pounds. These

differences in cases make it difficult to compare with each other. However, they are useful for comparisons with the random cases.

For the more loosely-constrained problem, there are sufficient facilities available that KP does not need to add any. However, KP always uses all 13 facilities -- even for those cases where the minimum number of facilities is only 4 or 5. The initial seeds for KP are actually centroids of clusters of the customers which indicates that they are pretty well evenly distributed across the customer locations. Since there is enough capacity that most customers can probably be assigned to the closest seed, it makes sense that KP would use all available facilities.

The APCI cases also use all 13 facilities, even though their total demand to total capacity ratios are very low (below 0.50). However, since they are non-random, their demands are not evenly distributed along a given range. There may be two customers fairly close together that are assigned to different facilities because there is not sufficient capacity to handle their demand. This phenomenon could also happen in the random cases, but there is a greater likelihood of it happening with the APCI cases.

For this analysis, the "best" parameters don't need to be determined since adding more facilities will always improve the results. KP appears to use all available facilities, though there is a point where it would not. In future analyses, the seed initialization procedure will be provided with 13 output nodes and capacity of 250 lbs for the random cases and 1.2 million lbs for the APCI cases.

5.3 Order of Input Pattern Presentation

In order to determine the "best" ordering of demand for KP, the other parameters were fixed at the following values:

- capacity for random cases (50, 75 and 100 customers): 250 lbs
- capacity for APCI cases (300 customers): 1,200,000 lbs
- initial weights on output nodes: equal to median of customer latitudes and longitudes
- initial learning rate: 1.0
- learning rate decay: geometric (0.99)
- number of cycles: 100
- number of facilities specified: 13

The average total Euclidean distance for each problem type and the average of the maximum cluster distance are displayed in Table 8 for the three scenarios.

Table 8. Comparison of different orderings of demand in KP*.

Number of Custs	Random Demand Order	Increasing Demand Order	Decreasing Demand Order
50	268 / 51	274 / 43	302 / 49
75	477 / 67	490 / 85	479 / 77
100	688 / 96	679 / 93	691 / 86
300	561 / 157	459 / 101	464 / 91

* Results represent average of total Euclidean distance / average of maximum cluster distance for each problem type.

Although it is difficult to discern which ordering of demand works best for the randomly generated problems (the results are not significantly different), it is clear that some ordering of demand (either increasing or decreasing) works best for the APCI problems. There isn't any statistical

difference between the increasing demand and decreasing demand scenarios for the APCI problems. However, looking at the maximum cluster distance, there is a statistical difference between the distance for decreasing order of demand versus random demand, where there is not a statistical difference between increasing order of demand versus random demand.

From the previous analysis of the number of output nodes, it is clear that the order in which the customer input patterns are presented will have some effect, especially for the APCI cases where there are some customers with very large demand. More of an impact on CLOC is expected. The seed initialization procedure of KP does not consider capacities, and KP does its own sorting of customer records anyway (based on regret). The effect on KP is indirect. It is really the order of customer input patterns and thus the effect on the weight update that causes different seeds to be produced by the seed initialization procedure.

In the previous capacitated analyses where 13 facilities were provided, KP used all of them. However, in this analysis, one APCI case used only 12 facilities when customer data was presented in an increasing or decreasing order of demand. Thus, by juggling the demand around, the number of facilities used was reduced. Also, for those cases where an improvement in total distance was made, there was almost always a corresponding improvement in the maximum cluster distance.

Based on these results, a random presentation of customer input patterns for the randomly generated problems will be used in future analyses. Since there is not a statistical difference between random ordering and non-random ordering, sorting the customer input patterns is not warranted. However, sorting the input patterns is warranted for the APCI

problems. At this point, it appears that a decreasing order of demand is appropriate for future analyses of the APCI problems based on the results of individual cases. It may become apparent when the final parameters are tested.

5.4 Initial Weight Vectors

Finally, in order to determine the best initial weights for the seed initialization procedure of KP, the other parameters were fixed at the following values:

- capacity for random cases (50, 75 and 100 customers): 250 lbs
- capacity for APCI cases (300 customers): 1,200,000 lbs
- initial learning rate: 1.0
- learning rate decay: geometric (0.99)
- number of cycles: 100
- number of facilities specified: 13
- ordering of demand for random cases: random
- ordering of demand for APCI cases: decreasing

The average total Euclidean distance for each problem type and the average of the maximum cluster distance are displayed in Table 9 for the three scenarios: 1) all weight vectors initialized to customer median latitude and longitude, 2) output nodes divided into 4 sets where a combination of relatively high and relatively low latitude and longitude are used as the weight vectors and 3) weight vectors initialized where customers are more densely located (APCI cases only).

Table 9. Comparison of different initial weight vectors in the seed initialization procedure of KP*.

Number of Custs	Initial Weights as Median Lat and Long	Initial Weights in Quadrants	Initial Weights where Customers Located
50	268 / 51	268 / 49	N/A
75	477 / 67	477 / 68	N/A
100	688 / 96	687 / 94	N/A
300	464 / 91	460 / 84	464 / 91

* Results represent average of total Euclidean distance / average of maximum cluster distance for each problem type.

For the randomly-generated problems and APCI problems, there is not any statistical difference between using the median of customer latitudes and longitudes as the initial weights for all output nodes versus breaking the customer region into four areas and assigning initial weights within those four areas. Since the APCI problems are non-random, initializing the weights to areas where customers are more densely located was also tested. The mean for these results is also not significantly different from the results using the median latitude and longitude or initial weights in quadrants.

The seed initialization procedure of KP ultimately produces clusters of customers and it is the centroid of each cluster that is the initial seed for KP. The seed initialization procedure just uses the initial weights that have been analyzed as a starting point. KP will adjust the initial seeds produced in its processing. Therefore, it seems intuitive that different initial weights will not have a large impact on the results of KP. The conclusion is made that all initial weights assigned to the same value -- median of the customer latitudes and longitudes -- is sufficient for KP.

5.5 Best Parameters for Seed Initialization Procedure of KP

In summary, the following parameters work best in the seed initialization procedure of KP:

- initial learning rate: 1.0
- learning rate decay: geometric (0.99)
- number of cycles: 100
- number of facilities specified: 13
- capacity for random cases (50, 75 and 100 customers): 250 lbs
- capacity for APCI cases (300 customers): 1,200,000 lbs
- ordering of demand for random cases: random
- ordering of demand for APCI cases: decreasing
- initial weights on output nodes: equal to median of customer latitudes and longitudes

These parameter settings produce results displayed in Table 10.

Table 10. Results produced from "best" parameters determined for the seed initialization procedure of KP*.

Number of Custs	Using Best Parameters
50	268 / 51
75	477 / 67
100	688 / 96
300	464 / 91

* Results represent average of total Euclidean distance / average of maximum cluster distance for each problem type.

6.0 CLOC Algorithm Discussion

Laura Burke has developed a neural method for solving Euclidean location and location-allocation problems. CLOC (Competitive LOCation) is a competitive learning approach with a slightly modified weight update to

handle the Euclidean *location* problem as opposed to a clustering problem (Burke, 1994). Burke has found promising results with the testing she's performed.

The algorithm has been modified slightly to handle the capacitated location-allocation problem. There are 3 main steps to the CLOC algorithm. First, initialize all weight vectors, the learning rate and the decay in the learning rate. Then, present patterns to the network, where a pattern consists of customer latitude and longitude and demand. The output nodes of the network are capacitated. The output node that "wins" the competition is the node that has the minimum Euclidean distance to the customer with enough available capacity to satisfy the demand. Finally, update the weight vector and the learning rate of the winning output node. Repeat these steps until the maximum number of cycles has been reached.

7.0 CLOC Implementation

The next sections describe the implementation of CLOC in more detail. The test problems used are then described and finally results of the analyses are discussed.

CLOC was written by Burke and coded in Microsoft FORTRAN. The code was modified slightly to allow for capacities on the output nodes and demands on the input patterns. More details on the steps in the algorithm appear below:

1) Initialize the weight vector for all output nodes where the number of output nodes is fixed *a priori*.

2) Initialize other parameters such as learning rate, decay in learning rate at each cycle and the number of cycles.

3) Read an input pattern (X_1, X_2, demand) where X_1 is customer latitude, X_2 is customer longitude and demand is customer demand.

4) The output node that wins the competition is the output node closest in the Euclidean sense to input pattern i that has capacity - demand ≥ 0. Allowing initial capacity violations in the algorithm and restricting the distance between customer and output node will be discussed in Sections 8.5 and 8.6, respectively.

5) Update the weight vector and the learning rate for the winning node J only as follows:

$$W_{iJ}^{new} = W_{iJ}^{old} + \eta_J(X_{iJ} - W_{iJ}^{old}) / |\mathbf{X_I} - \mathbf{W_J}|$$

$$\eta_J = \eta_J * \text{decay term}$$

$$\text{capacity}_J = \text{capacity}_J - \text{demand}_i$$

6) Return to Step 3) and repeat for all input patterns. When all input patterns are read, increase the cycle counter and read through the input patterns again. Continue until the maximum cycle counter value is reached.

A copy of the code for CLOC can be found with Dr. Burke in the Industrial Engineering Department.

8.0 CLOC Test Problems

The same 20 test cases that were used in testing KP were used to test CLOC. Again, the goal is to find what "parameters" work best for CLOC. Using the same value for each parameter tested in the seed initialization

procedure of KP and CLOC was attempted . However, there were some cases where different parameter values were used due to results observed. Below are the parameters that this research focused on tuning for CLOC:

1) learning rate and learning rate decay
2) number of output nodes and the capacity on those nodes
3) order of presentation of input patterns
4) initial weight vectors
5) initial capacity violations allowed
6) maximum cluster distance tolerance

8.1 Learning Rate and Learning Rate Decay

Again, a geometric decrease in learning rate will be compared with a linear decrease. An initial learning rate close to 1 and a geometric decrease in learning rate close to 1 after each cycle should provide good results. Will a linear decrease work as well or better? The following scenarios were tested:

1) initial learning rate of 1.0, geometric decrease after each cycle of 0.99, 100 iterations
2) initial learning rate of 1.0, linear decrease after each cycle of 0.005, 20 iterations

In the results, the reasons for using a different scenario in testing the linear decrease in learning rate will be discussed.

8.2 Number of Output Nodes and Capacities on those Nodes

The same two scenarios used in testing KP will be used:

1) the minimum number of output nodes required
2) more-than-enough output nodes

It will be interesting to see how many output nodes are used when testing 2) above. Burke discusses the fact that a neural network approach will determine how many output nodes to use, of the fixed number provided (Burke, 1994). The total demand to total capacity ratio will be watched closely to see if any conclusions can be drawn about its effect.

8.3 Order of Input Pattern Presentation

The same scenarios for presenting customer records were used as with KP: random order, increasing order of demand and decreasing order of demand. Of interest are 1) the final solution (especially total Euclidean distance) and 2) the number of output nodes formed, especially in the tightly-constrained scenario.

8.4 Initial Weight Vectors

The same 3 scenarios were used as tested with KP:

1) All output nodes initialized to the same weight vector -- the median latitude and longitude of all customer locations
2) Output nodes divided into 4 sets where the sets are: high/high, high/low, low/high, low/low. High/high indicates the latitude and longitude are both relatively high, high/low means the latitude is relatively high, but the longitude is relatively low, etc.
3) For the Air Products data only, initial weight vectors in the area where customers are more densely located were tested.

How much of an impact do different initial weights have on the final solution? These initial weights are the initial seeds that CLOC starts with. Again, they need to be chosen carefully.

8.5 Allowing Initial Capacity Violations

Allowing initial capacity violations is a parameter unique to CLOC, since the seed initialization procedure of KP does not consider customer demands and facility capacities. This parameter is used in Step 4) of the CLOC algorithm in Section 7.0. Step 4) becomes:

> 4) The output node that wins the competition is the output node closest in the Euclidean sense to input pattern i that has remaining capacity - demand $\geq X$ (where X is the capacity violation parameter.)

X is initialized to a negative value and is gradually updated after each cycle so it moves closer and closer to zero. Initial values of X are chosen to allow several customers to violate the capacity. In other words, a value of -50 for X will allow two customers to violate the capacity if their average demand is 25. However, finding appropriate values for this parameter involves trial and error.

What effect does this parameter have on the results? Is CLOC better able to assign customers to seeds if it doesn't have to worry about violating capacities initially? Does allowing initial capacity violations affect the results more for tightly-constrained problems (where the total demand to total capacity ratio is high)?

8.6 Cluster Distance Tolerance

A tolerance variable on the cluster distance is another parameter unique to CLOC. It is designed to prevent assigning a customer to a seed where the associated Euclidean distance is above a specified value. It is also used in Step 4) of the CLOC algorithm. Step 4) becomes:

4) Find the output node that is closest in the Euclidean sense to input pattern i with capacity - demand ≥ 0. If the distance between input pattern i and that output node is greater than the tolerance, assign input pattern i to an unused output node. If there is not an unused output node, assign the input pattern to the output node found to be closest in the Euclidean sense.

The tolerance variable is another parameter that involves trial and error to find an appropriate value. A starting point would be to look at previous results and use a value of the same magnitude as the maximum distance from a customer to its assigned location. Obviously, if capacity is tight, there will be some cases where this tolerance cannot be adhered to. Therefore, final results may have distances from a customer to its assigned seed greater than the tolerance value.

9.0 CLOC Computational Results

In the following sections, details from testing scenarios for the various parameters for CLOC described above are discussed. As with the results from KP, the parameters fixed and the values at which they are fixed are stated. Results are also discussed, in terms of what value for each parameter worked best, as well as general observations made while testing that particular parameter. Following the discussion of the results, "best" results from KP will be compared with "best" results from CLOC. The computational performance of the two algorithms will also be compared.

9.1 Learning Rate and Learning Rate Decay

In determining the "best" learning rate and learning rate decay to use for CLOC, the other parameters were fixed at the following values:

- number of output nodes for 50-customer cases: 5
- number of output nodes for 75-customer cases: 8
- number of output nodes for 100 customer cases: 10
- number of output nodes for 300 customer cases: 10
- capacity on each output node: equal to total demand
- initial weights on output nodes: equal to median of customer latitudes and longitudes

These parameter values are the same as the values used in the seed initialization procedure of KP. Again, this scenario is essentially non-capacitated because of the number of output nodes and the capacities on those output nodes.

As a starting point, CLOC results are compared with a k-means clustering algorithm as a benchmark, as was done when testing KP. The results are displayed in Table 11 along with the results from the two learning rate scenarios. The values shown represent the average total Euclidean distance (between each weight vector formed and its assigned customers) and the average of the maximum cluster distance for each problem type.

Table 11. Comparison of k-means clustering and CLOC with various values of learning rate and learning rate decay*.

Number of Custs	k-means clustering	1.0 initial η *0.99 decay 100 cycles	1.0 initial η -0.005 decay 20 cycles
50	564 / 186	557 / 197	586 / 205
75	691 / 149	749 / 217	821 / 206
100	849 / 155	890 / 203	1057 / 262
300	679 / 216	658 / 210	N/A

* Results represent average of total Euclidean distance / average of maximum cluster distance for each problem type.

The table shows that CLOC, with the geometric decrease, sometimes is better than k-means and sometimes is not. In looking at the individual results, 60% of the CLOC cases performed better than k-means clustering. However, no statistical conclusions can be drawn.

From the results, it is apparent that CLOC with a geometric decrease in learning rate performs better than a linear decrease in learning rate. The average distance for CLOC with a linear decrease is significantly different than k-means in the 75 customer and 100 customer cases. Also for the 100 customer case, CLOC with a linear decrease in learning rate is significantly different than CLOC with a geometric decrease.

The linear decrease in learning rate produced strange results. The total distance would gradually decrease over several iterations, then would steadily increase to very large numbers. When testing the APCI problems, only four cycles could be used before the results became unstable, thus the N/A in the table above.

In future analyses, an initial learning rate of 1.0, followed by a 99% reduction in learning rate after each cycle for 100 cycles will be used. Again,

Burke (Burke, 1994) has found good results with such values. In addition, the results are not significantly different from k-means.

9.2 Number of Output Nodes and Capacities on those Nodes

In analyzing the number of output nodes to use in CLOC, the other parameters were fixed at the following values:

- capacity for random cases (50, 75 and 100 customers): 250 lbs
- capacity for APCI cases (300 customers): 1,200,000 lbs
- initial weights on output nodes: equal to median of customer latitudes and longitudes
- initial learning rate: 1.0
- learning rate decay: geometric (0.99)
- number of cycles: 100

These parameter values are the same as the parameters used in the seed initialization procedure of KP. The two scenarios tested are the same as well: a tightly-constrained problem and a loosely-constrained problem. The results for this analysis appear in Table 12.

Table 12. Comparison of tightly-constrained case (Scenario 1) and loosely-constrained case (Scenario 2) based on total demand to total capacity ratio*.

# of Custs	Scenario 1	Ratio for Clusters Spec.	Ratio for Clusters Formed	Scenario 2	Ratio for Clusters Spec.	Ratio for Clusters Formed
50	604 / 187	0.78 - 0.96	0.78 - 0.96	380 / 110	0.24 - 0.44	0.31 - 0.60
75	721 / 132	0.91 - 0.99	0.87 - 0.94	595 / 109	0.42 - 0.67	0.42 - 0.83
100	996 / 156	0.93 - 0.99	0.91 - 0.97	751 / 121	0.45 - 0.84	0.49 - 0.84
300	1852 / 1133	0.46 - 0.93	0.46 - 0.93	564 / 186	0.04 - 0.50	0.04 - 0.50

* Results represent average of total Euclidean distance / average of maximum cluster distance for each problem type.

Once again, the loosely-constrained cases are obviously better than the tightly-constrained cases because there are more facilities over which to spread the customers' demand, thus creating tighter clusters. The total demand to total capacity ratios for the number of facilities specified and the number of facilities formed is also indicated. For the minimum number of output nodes case, additional facilities were required for test problems with a total demand to total capacity ratio of 96% and greater. However, there was a 96% and a 97% ratio that didn't require extra facilities.

For the loosely-constrained cases, 13 facilities are allowed. However, unlike KP, CLOC doesn't always use all 13 facilities. There is not any direct comparison between the total demand to total capacity ratio and the need for additional facilities. In the random problems where the ratio is high (0.71 and above), all 13 facilities are used. This is expected. But in the APCI problems where the ratio is significantly smaller, all 13 facilities are used in 80% of the problems. Since these APCI problems are not randomly

generated, the customers are not evenly distributed over the range of latitudes and longitudes. The additional facilities may be required to cover the "unusual" customers.

Once again, the "best" parameters don't need to be determined since adding more facilities will always improve the results. Thirteen facilities appears to be a reasonable number since CLOC will only use the facilities it needs anyway. In future analyses, CLOC will be provided with 13 output nodes and capacity of 250 lbs for the random cases and 1.2 million lbs for the APCI cases.

9.3 Order of Input Pattern Presentation

In order to determine the "best" ordering of demand for CLOC, the other parameters were fixed at the following values:

- capacity for random cases (50, 75 and 100 customers): 250 lbs
- capacity for APCI cases (300 customers): 1,200,000 lbs
- initial weights on output nodes: equal to median of customer latitudes and longitudes
- initial learning rate: 1.0
- learning rate decay: geometric (0.99)
- number of cycles: 100
- number of facilities specified: 13

The average total Euclidean distance for each problem type and the average of the maximum cluster distance are displayed in Table 13 for the three scenarios.

Table 13. Comparison of different orderings of demand in CLOC*.

Number of Custs	Random Demand Order	Increasing Demand Order	Decreasing Demand Order
50	380 / 110	393 / 118	397 / 110
75	595 / 109	572 / 121	543 / 98
100	751 / 121	742 / 114	711 / 100
300	564 / 186	458 / 120	463 / 113

* Results represent average of total Euclidean distance / average of maximum cluster distance for each problem type.

These results are similar to KP's results in that it's difficult to discern which ordering of demand works best for the randomly-generated problems. The results are not significantly different. However, it is clear that some ordering of demand (either increasing or decreasing) works best for the APCI problems. There isn't any statistical difference between the increasing demand and the decreasing demand scenarios for the APCI problems.

Looking at the maximum cluster distance, once again, similar to KP results, there is a statistical difference between the maximum cluster distance for decreasing order of demand versus random demand. There is not such a difference between increasing order of demand versus random demand.

For the random ordering of demand, it was observed that 13 facilities were used when the total demand to total capacity ratio is relatively high. All 13 facilities are also used in 80% of the APCI problems. Because these problems are non-random, more facilities are required to handle the "unusual" customers.

Ordering the demand has an effect on the number of facilities required. For the problems where increasing demand was used, sometimes fewer

facilities are required and sometimes more facilities are required. The same is true for the problems using decreasing demand.

The APCI cases, once again, are a bit different. For the increasing demand problems, all problems require 13 facilities. One problem even requires 15 facilities -- the problem with the largest demand to capacity ratio. It is also the problem where roughly 6% of the customers have over 60% of the demand. For the decreasing demand problems, all cases require 13 facilities.

For the randomly-generated problems, a random ordering of demand will be used. However, a decreasing order of demand will be used for the APCI problems because of the significant difference in results. The other orderings of demand will still be considered when the remaining parameters are analyzed. It may be that, when combined with other parameters, a different ordering of demand works better.

9.4 Initial Weight Vectors

In order to determine the best initial weights for CLOC, the other parameters were fixed at the following values:

- capacity for random cases (50, 75 and 100 customers): 250 lbs
- capacity for APCI cases (300 customers): 1,200,000 lbs
- initial learning rate: 1.0
- learning rate decay: geometric (0.99)
- number of cycles: 100
- number of facilities specified: 13

The average total Euclidean distance for each problem type is displayed in Table 14 for random ordering of demand, Table 15 for increasing order of

demand and Table 16 for decreasing order of demand. Results for the three scenarios are shown: 1) all weight vectors initialized to customer median latitude and longitude, 2) output nodes divided into 4 sets where a combination of relatively high and relatively low latitude and longitude are used as the weight vectors and 3) weight vectors initialized where customers are more densely located (APCI cases only).

Table 14. Comparison of different initial weight vectors in CLOC with random ordering of demand*.

Number of Custs	Initial Weights as Median Lat and Long	Initial Weights in Quadrants	Initial Weights where Customers Located
50	380	285	N/A
75	595	502	N/A
100	751	709	N/A
300	564	574	556

* Results represent average of total Euclidean distance for each problem type.

When using a random order of demand, the difference between the average total distance for the median latitude and longitude as the initial weights and initializing weights in quadrants is statistically significant for the 50 and 75 customer problems. The difference in averages is not significantly different for the 100 and 300 customer problems. In addition, for the APCI cases, there isn't any statistical difference between the various methods for initializing weights.

Table 15. Comparison of different initial weight vectors in CLOC with increasing order of demand*.

Number of Custs	Initial Weights as Median Lat and Long	Initial Weights in Quadrants	Initial Weights where Customers Located
50	393	274	N/A
75	572	505	N/A
100	742	727	N/A
300	458	465	474

* Results represent average of total Euclidean distance for each problem type.

When using an increasing order of demand, the difference between the average total distance when the median latitude and longitude are the initial weights and initializing weights in quadrants is also statistically significant for the 50 and 75 customer problems. The difference in averages is not significantly different for the 100 and 300 customer problems. Once again, for the APCI cases, there isn't any statistical difference between the various methods for initializing weights.

Table 16. Comparison of different initial weight vectors in CLOC with decreasing order of demand*.

Number of Custs	Initial Weights as Median Lat and Long	Initial Weights in Quadrants	Initial Weights where Customers Located
50	397	287	N/A
75	543	506	N/A
100	711	724	N/A
300	463	476	457

* Results represent average of total Euclidean distance for each problem type.

When using a decreasing order of demand, the difference between the average total distance when the median latitude and longitude are the initial

weights and initializing weights in quadrants is only statistically significant for the 50 customer problems. The difference in averages is not significantly different for the 75, 100 and 300 customer problems. Again, for the APCI cases, there isn't any statistical difference between the various methods for initializing weights.

The results show that using different initial weights does not always produce a significant difference in the average distances. However, the individual results do indicate that using initial weights located in four areas of the customer region has a positive impact on the results. For the randomly-generated problems, 87% of the problems with random demand are better with initial weights in quadrants. 87% of the problems with increasing demand are better and 73% of the problems with decreasing demand are better with initial weights in quadrants. Therefore, initial weights assigned in quadrants will be used in future analyses.

For the APCI problems, initial weights in quadrants works best in two cases, initial weights where customers are densely located works best in two cases and the other case produces the same results in both scenarios. However, because the average of the distances when using initial weights where customers are densely located is smaller than initial weights in quadrants, using these more finely-tuned initial weights may be considered in future analyses.

The results observed when analyzing the presentation of input patterns still hold true for this analysis. For the randomly-generated problems, no one method works better than the others. A random ordering of demand for these random problems will continue to be used. For the APCI problems, some ordering of the demand is better than random ordering.

Again, a decreasing order of demand may be used, but the effect of the last two parameters to analyze will first be examined.

9.5 Allowing Initial Capacity Violations

In order to determine the best initial weights for CLOC, the other parameters were fixed at the following values:

- capacity for random cases (50, 75 and 100 customers): 250 lbs
- capacity for APCI cases (300 customers): 1,200,000 lbs
- initial learning rate: 1.0
- learning rate decay: geometric (0.99)
- number of cycles: 100
- number of facilities specified: 13

In this analysis, a combination of initial weights and presentation of input patterns is examined while trying to determine whether or not to allow initial capacity violations.

In CLOC, there is a parameter which will allow capacities on facilities to go negative early on in the processing. The idea is that CLOC may possibly come up with better facility locations and assignments of customers to facilities if the capacity constraint isn't so tight initially. In the analysis, it was found that there isn't a particular value for this parameter that always works better than all other values. The value may vary problem by problem.

In Table 17, a comparison of results between allowing initial capacity violations and not allowing violations is displayed. For the randomly-generated problems, a random order of demand and initial weights located in quadrants is used. For the APCI problems, a decreasing order of demand

with the same initial weights as mentioned above is used, as well as initializing weights where customers are densely located.

Table 17. Comparison between allowing initial capacity violations and not allowing violations in CLOC*.

Number of Custs	Allow Initial Capacity Violations	No Initial Capacity Violations
50	294	285
75	495	502
100	746	709
300	492	476
300 [a]	457	459

* Results represent average of total Euclidean distance for each problem type.

[a] Using initial weights in areas of the customer region where customers are more densely located.

Allowing initial capacity violations does not have a significant impact on the results. For all problem types, the averages of the distances are not statistically different between the two scenarios tested. The same is true for the APCI problems using initial weights where customers are more densely located. In addition, no conclusions can be drawn about which initial weights work better. Since allowing initial capacity violations doesn't significantly improve the results, it will be ignored when considering the best parameters to use for CLOC.

9.6 Cluster Distance Tolerance

Finally, the last parameter analyzed in CLOC is the tolerance variable. In order to determine the effect of the tolerance variable and what value to use, the other parameters were fixed at the following values:

- capacity for random cases (50, 75 and 100 customers): 250 lbs
- capacity for APCI cases (300 customers): 1,200,000 lbs
- initial learning rate: 1.0
- learning rate decay: geometric (0.99)
- number of cycles: 100
- number of facilities specified: 13
- initial weights for randomly-generated problems: quadrants
- initial weights for APCI problems: where customers densely located
- order of demand for randomly-generated problems: random
- order of demand for APCI problems: decreasing

In order to determine what values to use for the tolerance variable, the distances from customers to their assigned facility in the results was determined. Different values that were in the neighborhood of the largest distance were chosen. Proceeding by trial and error, various values of the tolerance variable were tested, until some improvement in total distance was seen. Table 18 shows results for this analysis.

Table 18. Comparison of results with different tolerance values versus no tolerance variable in CLOC*.

Number of Custs	Tolerance = 10	Tolerance = 20	No Tolerance
50	281 / 61	N/A	284 / 60
75	479 / 81	N/A	501 / 86
100	N/A	700 / 110	709 / 93
300	441 / 85[a]	N/A	457 / 92

* Results represent average of total Euclidean distance / average of maximum cluster distance for each problem type.

[a] Tolerance = 5.

From the results it is clear that finding a value for the tolerance variable to work well with all problem types is not possible. Even within a class of problems, the same tolerance variable does not work best for all problems in that class. Although each problem type's results improve when a tolerance variable is used, the results are not significantly better. However, since the average distances have improved, each problem's best tolerance variable will be used when comparing CLOC with KP.

9.7 Best Parameters for CLOC

In summary, the following parameters work best in CLOC:

- initial learning rate: 1.0
- learning rate decay: geometric (0.99)
- number of cycles: 100
- number of facilities specified: 13
- capacity for random cases (50, 75 and 100 customers): 250 lbs
- capacity for APCI cases (300 customers): 1,200,000 lbs
- ordering of demand for random cases: random
- ordering of demand for APCI cases: decreasing
- initial weights for randomly-generated problems: quadrants
- initial weights for APCI problems: where customers densely located
- initial capacity violations: not allowed
- tolerance for 50-customer problems: 10
- tolerance for 75-customer problems: 10
- tolerance for 100-customer problems: 20
- tolerance for 300-customer problems: 5

These parameter settings produce results displayed in Table 19.

Table 19. Results produced from "best" parameters determined for CLOC*.

Number of Custs	Using Best Parameters
50	281 / 61
75	479 / 81
100	700 / 110
300	441 / 85

* Results represent average of total Euclidean distance / average of maximum cluster distance for each problem type.

10.0 Comparison of KP and CLOC

In this section, a comparison between KP and CLOC is made. First, the two algorithms are compared in terms of their computational performance. Next, the parameters that were determined to be best for the seed initialization procedure of KP and CLOC are compared. Then the final results produced when using the best parameters in KP and CLOC are compared. Finally, various observations made while testing KP and CLOC pertaining to how the two algorithms handled different problem types are discussed.

10.1 Performance

Both KP and CLOC were tested on a 486 / 66 MHz PC. Both algorithms are coded in Microsoft FORTRAN, including the seed initialization procedure of KP. CLOC's time to solve is 2 seconds for 50 customers, 4 seconds for 100 customers and 12 seconds for 300 customers. Obviously the time to solve is proportional to the number of customer input patterns, and also depends on the number of cycles. 100 cycles were used in almost all test cases.

Also, the number of output nodes (facilities) allowed didn't seem to have any effect on the time to solve. This means that how tightly the problem is constrained doesn't have any effect on performance. One could predict that a problem with 1000 customers, using 100 cycles, would take 40 seconds to solve on the same PC, since the performance seems to be proportional to the number of customer input patterns (and cycles).

With KP, the time for the algorithm to solve a particular problem cannot be estimated. Varying performance was observed depending on the specifics of the problem. For example, the 300 customer problem took between 18 seconds and 32 seconds to solve. The seed initialization portion of the algorithm consistently took 10 seconds to solve. Therefore, the actual KP code took between 8 and 22 seconds to solve. The case that ran for 32 seconds in total is one with larger demands.

Also, the performance for KP depends on the number of facilities allowed as well as the presentation of the customer input patterns. One of the more tightly constrained problems ran for 36 seconds. For the 100 customer case, performance is closer to CLOC -- 5 seconds where CLOC is 4 seconds. However, if there are any peculiarities in the data, time to solve is longer.

10.2 Parameters

For the most part, the parameters that are best for the seed initialization procedure of KP are the same as those that are best for CLOC:

- initial learning rate of 1.0
- 99% decrease in learning rate after each cycle
- 100 cycles

- 13 output nodes (capacity of 250 lbs for random problems, 1.2 million lbs for APCI problems)
- random problems use a random ordering of demand
- APCI problems use a decreasing order of demand

The only difference in parameters are the initial weights and those parameters that are unique to CLOC.

For the seed initialization procedure of KP, initial weights as the median of customer latitudes and longitudes are used. For CLOC, weights are initialized in quadrants for the randomly-generated problems and where customers are densely located for the APCI problems.

As was discussed earlier, the initial weights in the seed initialization phase of KP have an indirect effect on the results. The initial weights are gradually adjusted by the seed initialization procedure to produce initial seeds. KP then adjusts these initial seeds further. The effect in CLOC is more direct.

Finally, CLOC makes use of a tolerance variable in attempting to make the clusters of customers tighter. In the analysis, it was found that different problem types require different tolerance values in order to produce any improvement in total distance.

10.3 Comparison of Results

The results from the "best" parameters of the seed initialization of KP and the "best" parameters of CLOC appear in Table 20.

Table 20. Results produced from "best" parameters determined for seed initialization procedure of KP and CLOC*.

Number of Custs	KP	CLOC
50	268 / 51	281 / 61
75	477 / 67	479 / 81
100	688 / 96	700 / 110
300	464 / 91	441 / 85

* Results represent average of total Euclidean distance / average of maximum cluster distance for each problem type.

There is not a statistical difference between the best KP results and the best CLOC results. Based on this information, CLOC is a viable alternative to KP in solving the capacitated warehouse location-allocation problem.

In looking at the individual test results, CLOC has better results than KP in 47% of the randomly-generated problems. CLOC performs even better on the APCI problems -- 4 of the 5 problems have better results with CLOC than KP. For the remaining APCI problem, CLOC and KP produce equivalent results. From these results, it appears that CLOC and KP perform similarly for randomly-generated problems. However, CLOC does a better job when the problems are non-random.

10.4 Additional Observations

CLOC is better able to handle tightly-constrained problems compared to KP. These are problems where the total demand to total capacity ratio is high. In the analysis of the number of output nodes, KP had to add facilities for several problems where the minimum number of output nodes required was specified. KP added facilities for 8 test problems whereas CLOC only needed to add facilities for 3 test problems. Since it is reasonable to assume

that additional costs are associated with facilities, this characteristic of CLOC is quite advantageous.

When the problems are more loosely-constrained (the demand to capacity ratio is low), KP and CLOC also differ in the number of facilities that are used. KP tends to use all available facilities. CLOC doesn't always use all available facilities. This phenomenon is mainly due to the way the weights are initialized. CLOC tends to use more of the output nodes when the weights are initialized in quadrants of the customer region. When all weight vectors are initialized to the same value, as when using the median of customer latitudes and longitudes, some facilities remain unused.

Another observation is that the ordering of customer input patterns affects KP and CLOC differently. The ordering of demand affects KP indirectly. The KP algorithm re-sorts the customer patterns based on regret, so ordering the patterns won't directly affect KP. It will, to a small extent, affect the seeds produced by the seed initialization procedure due to the order in which the customers are assigned to facilities and thus how the weights are updated. The effect is more direct in CLOC, since it is CLOC that is updating the weights and making the customer assignments.

Finally, CLOC appears to handle real-world problems better than KP. Results are better and solution time is faster. CLOC's ability to handle these problems quickly and with good results makes this approach more desirable to use than KP. Solving real problems seldom involves data that is randomly distributed.

11.0 Summary and Discussion

CLOC is a viable approach for solving the capacitated warehouse location-allocation problem. It is an extremely fast algorithm, where its performance depends only on the size of the problem. Its computational performance seems to be superior to KP.

In CLOC, there are several parameters available for tuning. With careful selection of these parameters, CLOC produces results close to or better than KP. CLOC performs particularly well on real-world problems that are not random in nature.

CLOC also has the tendency to not use all available facilities. This feature is desirable when considering there are costs associated with using or building a facility. With its quick computational performance and good results, especially on real-world problems, CLOC is a viable alternative to KP in solving the capacitated warehouse location-allocation problem.

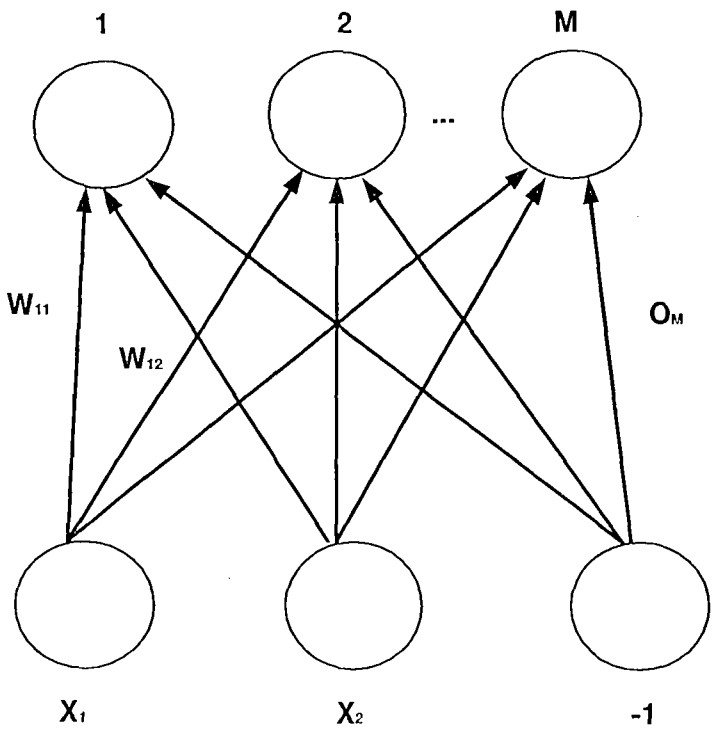

Figure 1. The neural network used in the seed initialization phase of Koskosidis and Powell's algorithm for CCP

X_1 is customer latitude, X_2 is customer longitude, W_{11} and W_{12} are weights, M is the number of facilities and θ_M is $(W_{1M}^2 + W_{2M}^2) / 2$.

References

Burke, L. (1994). "CLOC: An Adaptive Neural Method for Euclidean Location and Location-Allocation Problems", *Proceedings of the Third Industrial Engineering Research Conference*, pp. 261-266.

Koskosidis, Y. A. and W. B. Powell (1991). "Clustering Algorithms for Consolidation of Customer Orders into Vehicle Shipments", *Transportation Science 26B(5)*, pp. 365-379.

Potvin, J. (1994). "Integrating Operations Research and Neural Networks for Vehicle Routing", in The Impact of Emerging Technologies on Computer Science and Operations Research, S. Nash and A. Sofer (Eds), Kluwer Academic Publishers.

Vita

Jennifer Schmidt Chuss was born December 27, 1965 in Denville, New Jersey. She is the daughter of Robert and Patricia Schmidt. In 1988, Jennifer graduated from Carnegie Mellon University in Pittsburgh, Pennsylvania with a B.S. in Applied Mathematics (Operations Research) and a double major in Industrial Management. She spent 5 ½ years working in the Decisions Sciences group within MIS at Air Products and Chemicals, Inc. There, Jennifer was responsible for developing software applications and providing consulting services to help the Chemicals business areas improve their logistics decisions. Currently, she is part of a team developing a Knowledge Based System which will automatically generate Material Safety Data Sheets for the Chemicals Group's customers. On that team, she is responsible for knowledge elicitation and assisting in the development of the knowledge base.

END OF TITLE